



**UNIVERSITÄTSBIBLIOTHEK
BRAUNSCHWEIG**

Caroline - ein autonom fahrendes Fahrzeug im Stadtverkehr

**Thomas Form
Christian Berger
Karsten Cornelsen
Michael Doering
Jan Effertz
Tim Gülke**

**Kai Homeier
Alexander Movshyn
Tobias Nothdurft
Michael Sachse
Jörn-Marten Wille**

Braunschweig, 2011

Veröffentlicht: 08.03.2011

<http://www.digibib.tu-bs.de/?docid=00038404>

Caroline – ein autonom fahrendes Fahrzeug im Stadtverkehr

Thomas Form
Christian Berger
Karsten Cornelsen
Michael Doering
Jan Effertz
Tim Gülke
Kai Homeier
Alexander Movshyn
Tobias Nothdurft
Michael Sachse
Jörn-Marten Wille



Caroline – ein autonom fahrendes Fahrzeug im Stadtverkehr

Inhalt

1. Aufgabe und Zeitplan
2. Anforderungen
3. Team
4. Projektstruktur und Aufbau “Caroline”
5. Tests und Simulation
6. Prozesse
7. Zusammenfassung



Urban Challenge 2007

Aufgabe und Zeitplan

- 60 Meilen städtischer Kurs in weniger als 6:00h:
 - *Beachtung von Verkehrsregeln*
 - *Interaktion mit dynamischen Objekten (fließendem Verkehr)*
 - *Überquerung von Kreuzungen*
 - *Beachtung von Hindernissen*
 - *kurzfristige Änderungen in der Ziel-/Missionsplanung*
- **Terminplanung:**

– 13. April 2007	Video-Demonstration
– 11. Juni - 20. Juli 2007	Vorausscheid Site Visit
– 10. August 2007	Halbfinale Verkündigung Teilnehmer
– 21. Oktober - 31. Oktober 2007	Halbfinale National Qualification Event
– 03. November 2007	Finale



Urban Challenge 2007

Anforderungen – sicheres autonomes Fahren im Verkehr mit 20mph

- Autonomes durchfahren komplexer Missionen mit max. 5min Vorbereitungszeit
- Selbsttätiges Navigieren in Gebieten ohne vorgegebene Wegpunkte
- Dynamische Anpassung der Route bei unvorhergesehenen Hindernissen mit ausführen von 180° Kehren bei blockierten Strassen
- Erkennen von gelben und weißen Fahrbahnmarkierungen
- Situationsabhängige Geschwindigkeitswahl und einhalten von vorgegebenen Geschwindigkeitsbeschränkungen mit max. 30mph
- Richtiges Erkennen und einhalten der Fahrspur auf befestigten und unbefestigten Strassen auch in Gebieten ohne bzw. mit eingeschränkter GPS Unterstützung
- Einhalten vorgegebener Sicherheitsabstände zu vorausfahrenden Fahrzeugen bzw. in Stop-and-go Situationen
- Umfahren stehender Fahrzeuge und sicheres ein- und ausparken
- Spurwechsel beim Überholen ohne Gefährdung anderer Verkehrsteilnehmer und richtiges Verhalten vor Kreuzungen
- beachten von Vorfahrtregeln an Kreuzungen ohne unnötige Verweilen >10s.



Urban Challenge 2007

nicht geforderte Fähigkeiten

- Erkennen von Verkehrszeichen und Lichtsignalen durch die Sensorik des Fahrzeugs.
- Erkennen und ausweichen von Fußgängern.
- Geschwindigkeiten >30mph.
- Unbefestigte Wege, die ein geländegängiges Fahrzeug erfordern.

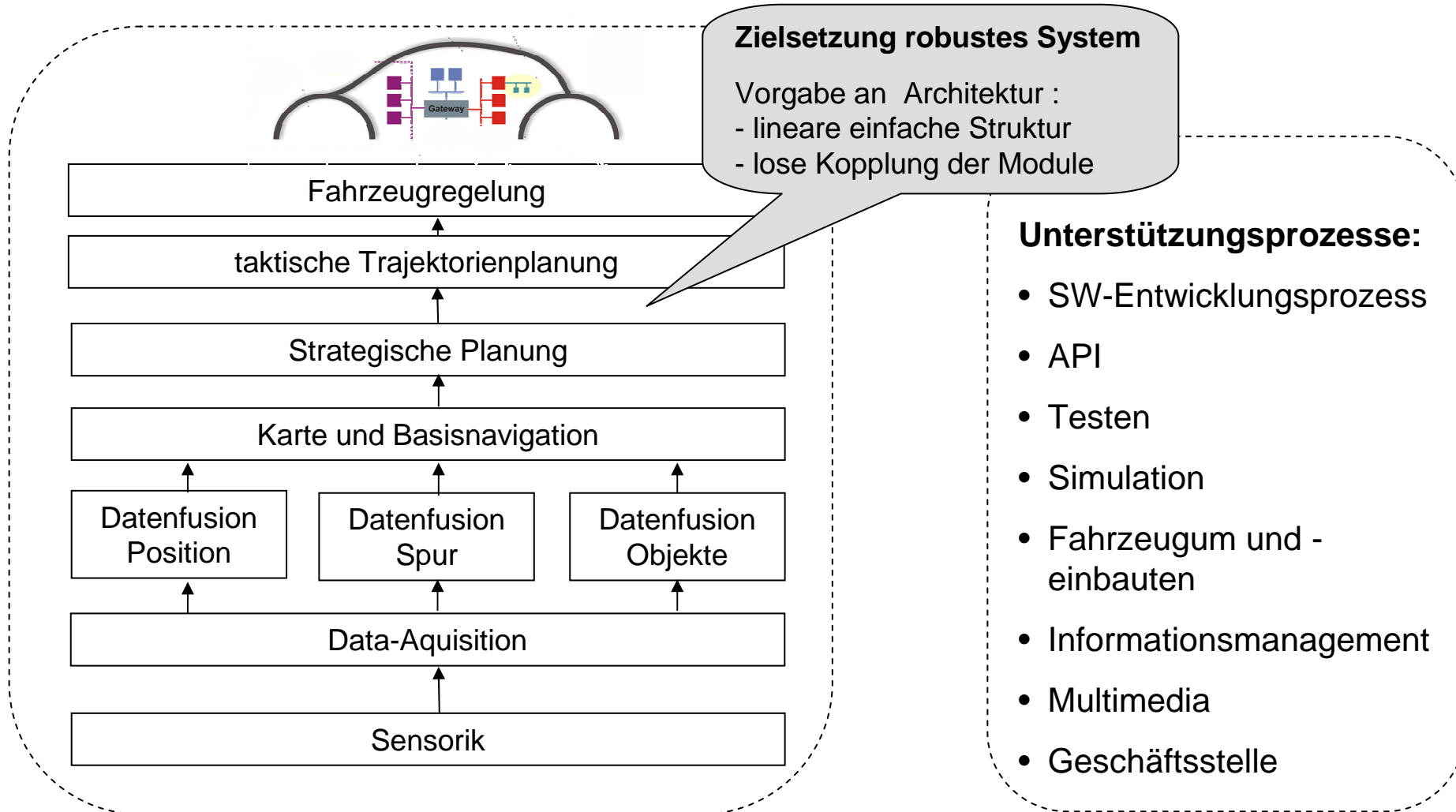


Das Team



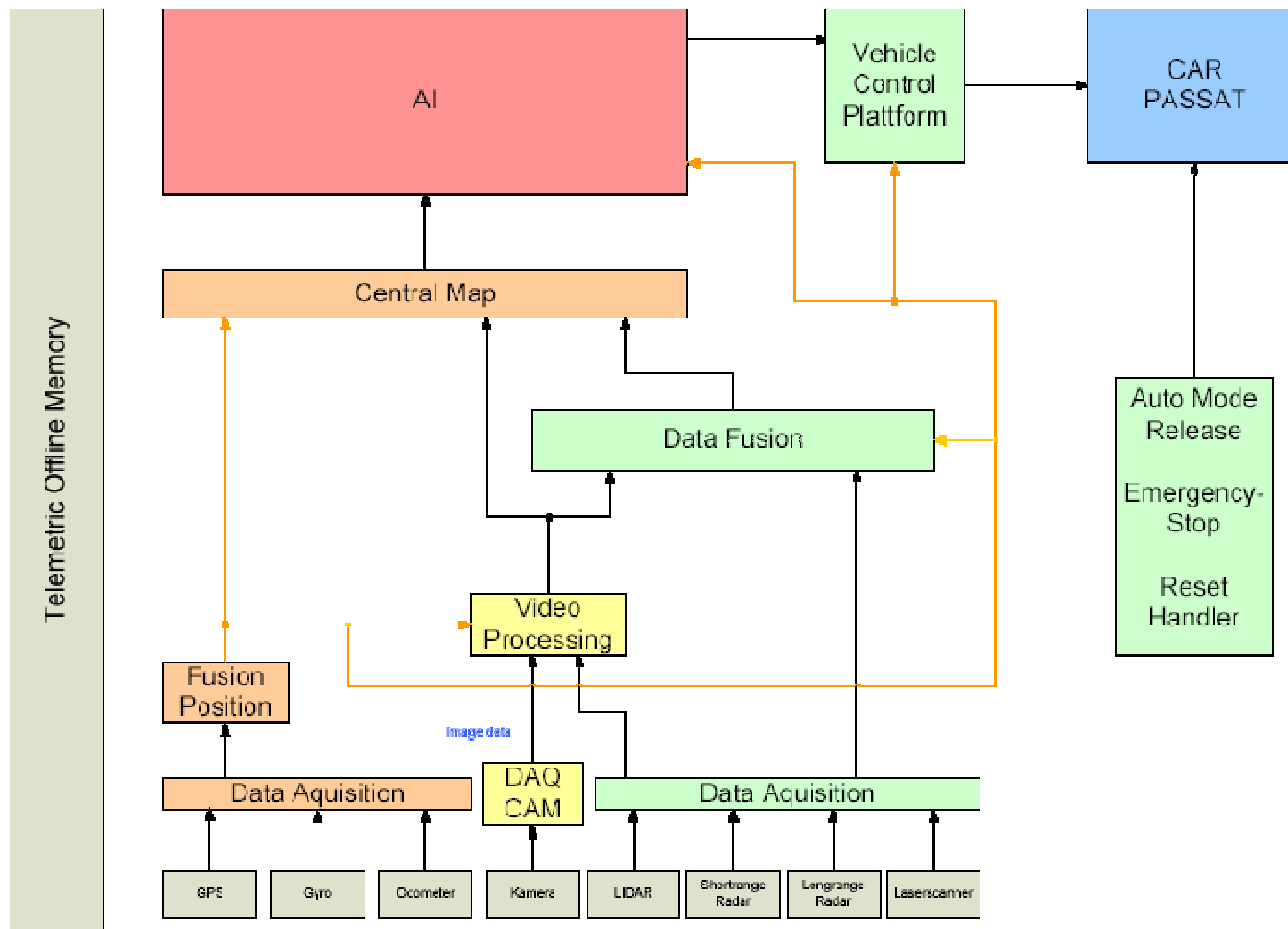
Projektstruktur und Aufbau "Caroline"

Strukturierung der Aufgaben



Projektstruktur und Aufbau "Caroline"

Feinstruktur

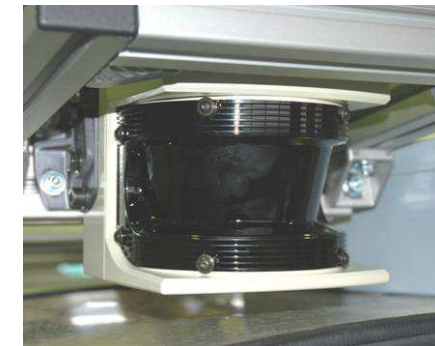
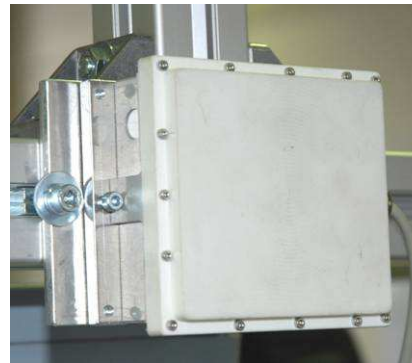


Fahrzeugumfelderkenennung und Sensorfusion

„Wer die Wahl hat, hat die Qual“

Verschiedenste Sensortypen am Markt:

- Radar
 - Laserscanner
 - Lidar
 - Standard-Vision
 - IR-Vision
 - Stereo-Vision
 - Ultraschall
-
- Jedes Messverfahren hat Stärken und Schwächen
 - Es existiert zur Zeit kein „All-Round“-Sensor



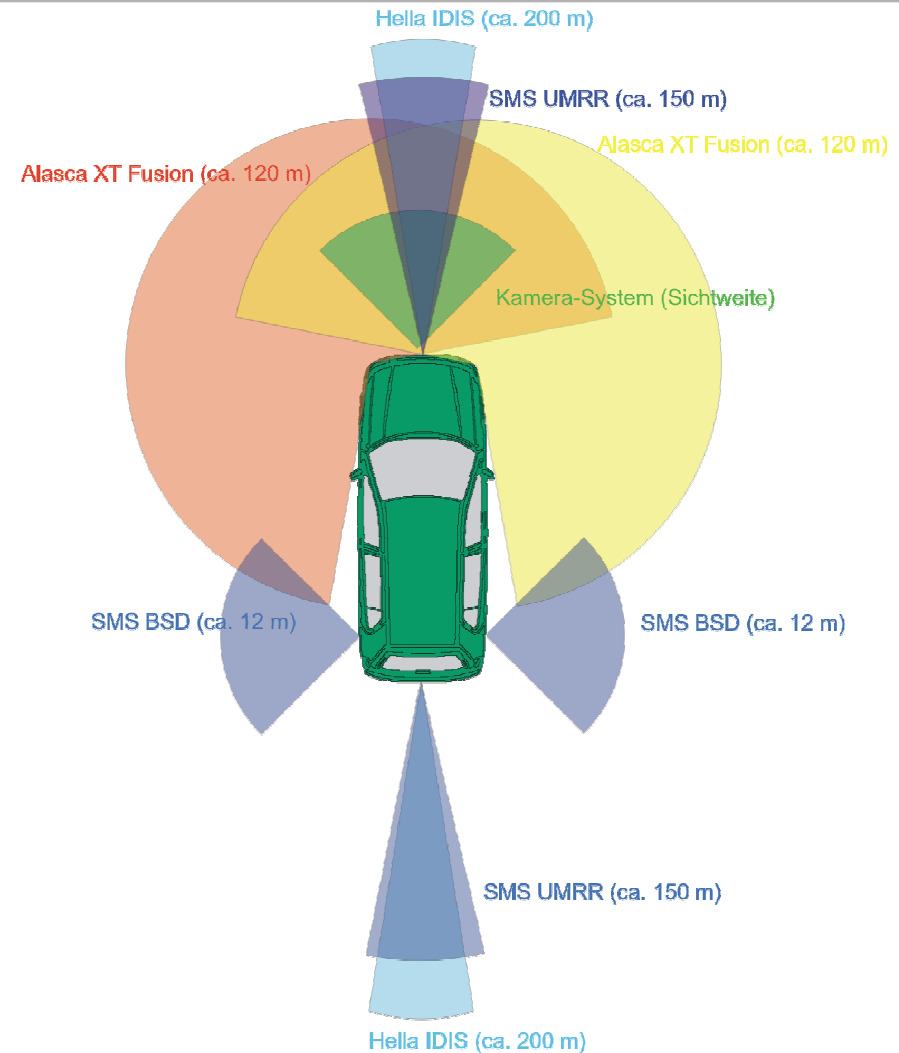
Fahrzeugumfeldererkennung und Sensorfusion

„Viele Augen sehen mehr“

Lösung: Multi-Sensor-Netzwerke

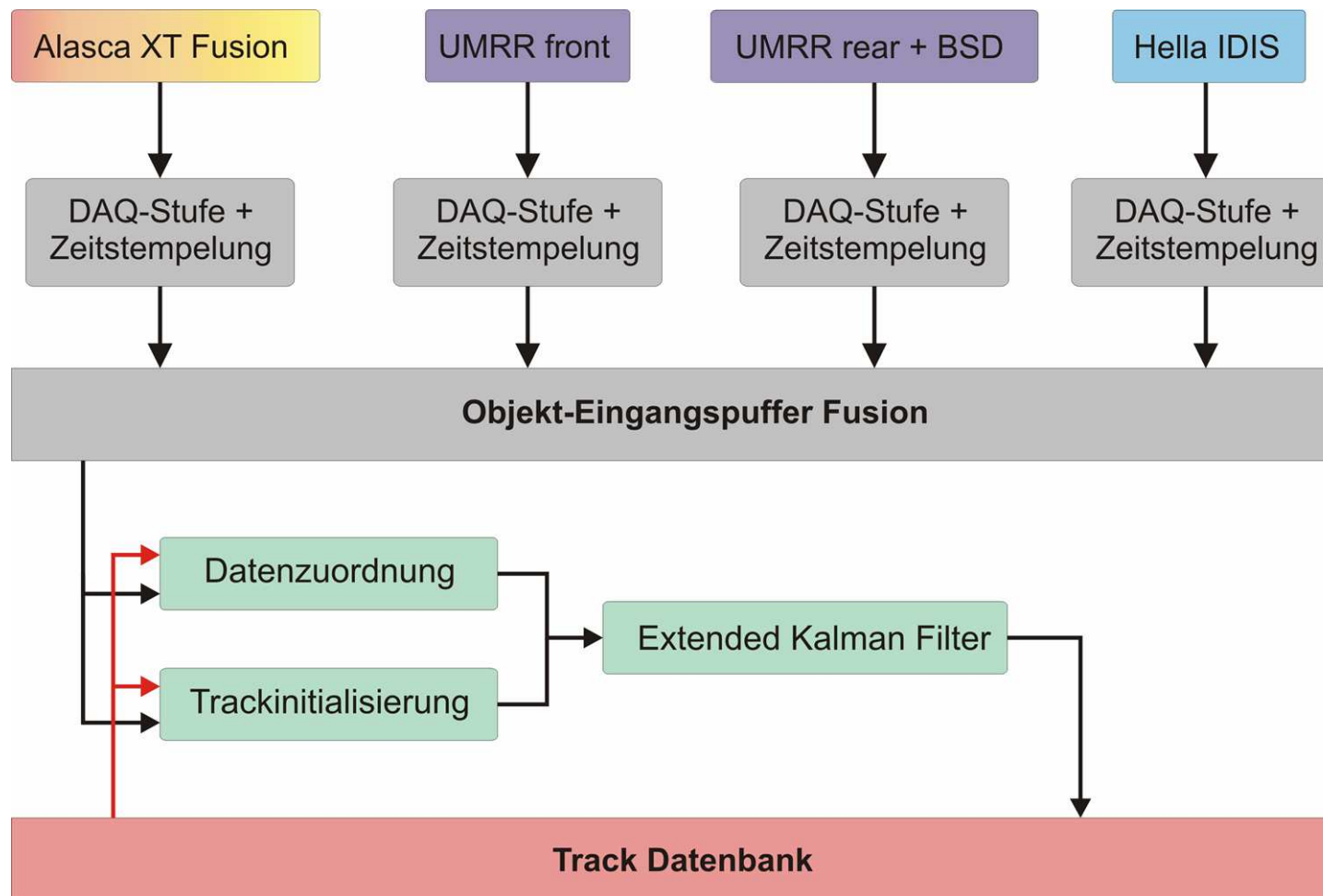
- Überlappende Beobachtungsbereiche
- Stärken und Schwächen kompensieren sich
- Redundanz (Sicherheitsaspekt)

Voraussetzung: **Fusion der Sensordaten**



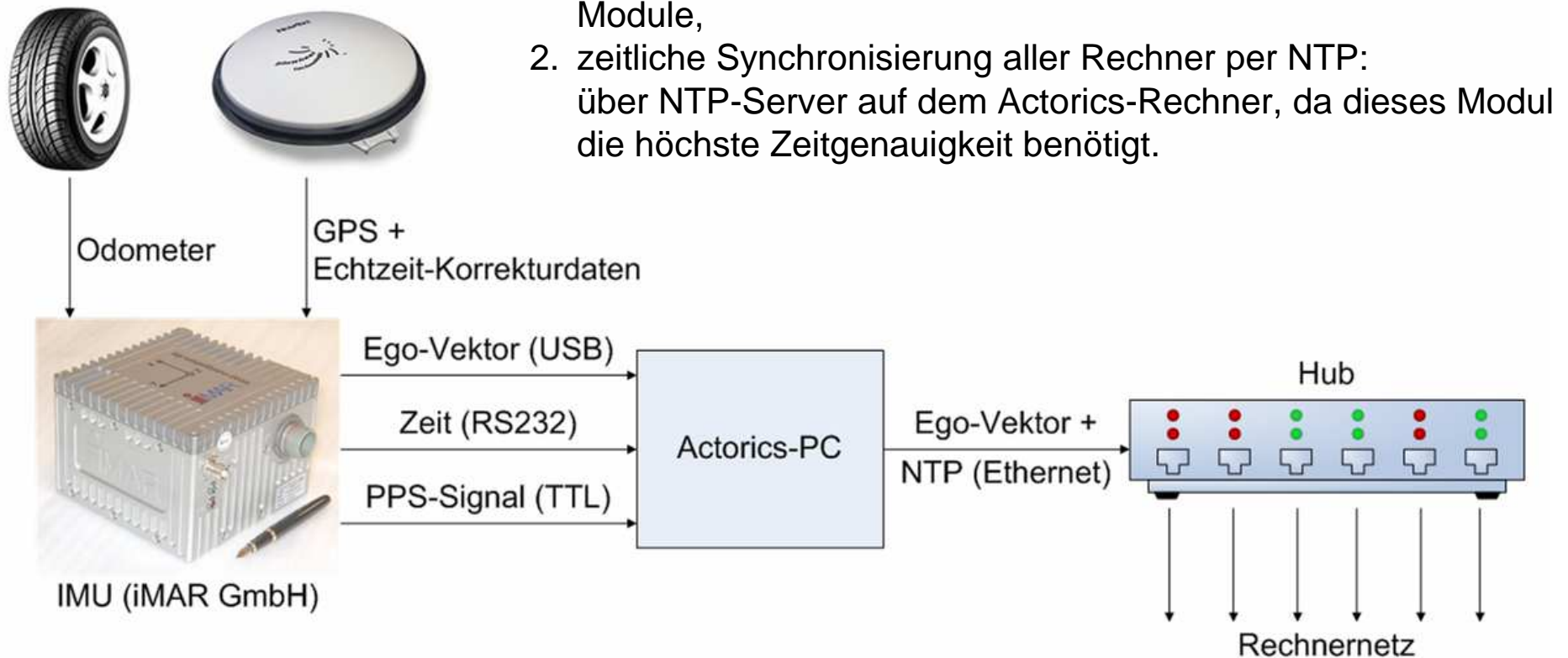
Fahrzeugumfeldererkennung und Sensorfusion

Konzept Tracking und Sensorfusion



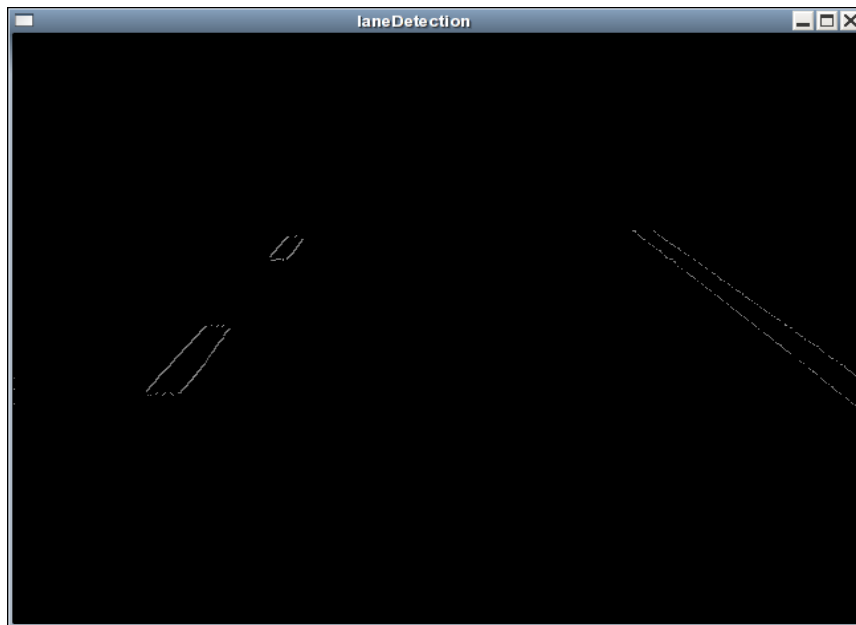
Sensorik

Inertial Measuring Unit

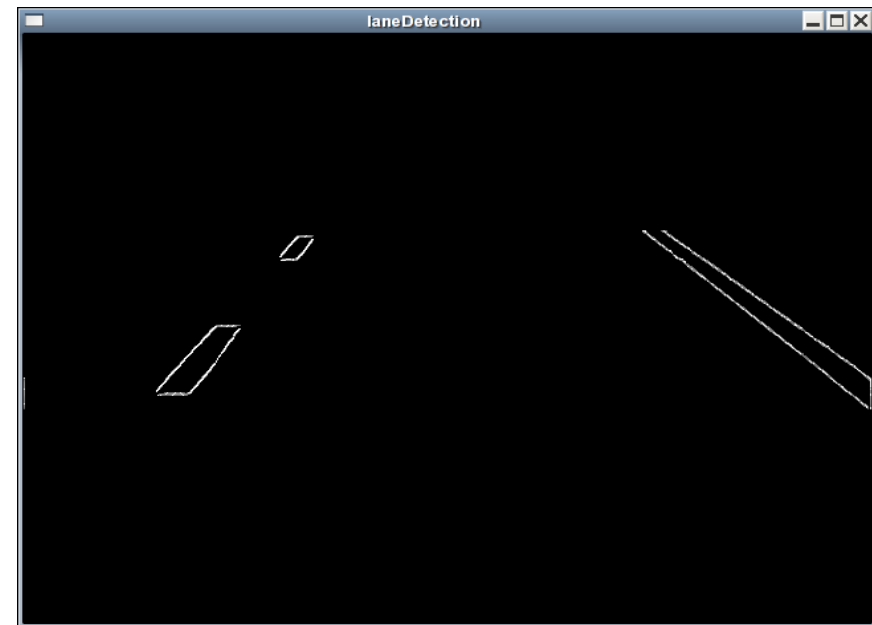


Spurerkennung – Initialisierung (1)

- Kantendetektion mit Hilfe des Canny-Filters



Sobel Gradient map



Canny-Edge result



Spurerkennung – Initialisierung (2)

Hough-Transformation für die Liniendetektion

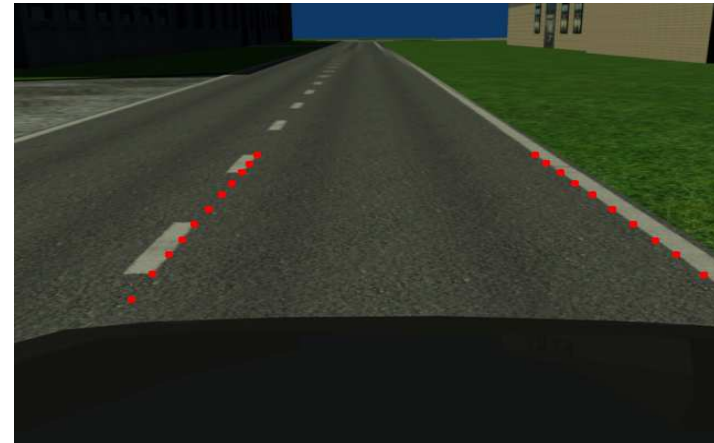
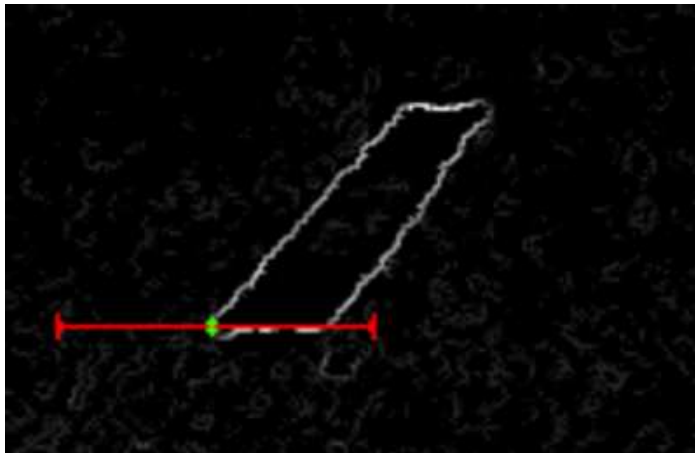


- Mit dem Einsatz der graphischen Hardware (GPU) dauert die Initialisierung nur etwa 0.1s:
 - Bildglättung und Canny-Edge 0.03 s
 - Hough Transformation 0.07s



Spurerkennung - Tracking

- Auf der Basis der aus der Initialisierung bekannten Richtungen der Fahrbahnmarkierungen wird ein Suchbereich entlang der x-Achse bestimmt.
- Die größten Gradienten in diesem Bereich liefern die gesuchten Fahrbahnmarkierungen.



Strategische Planung

Beispielszenario

Naive Ideen:

Direkt auf den Wegpunkt zufahren

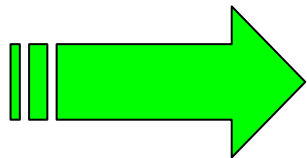
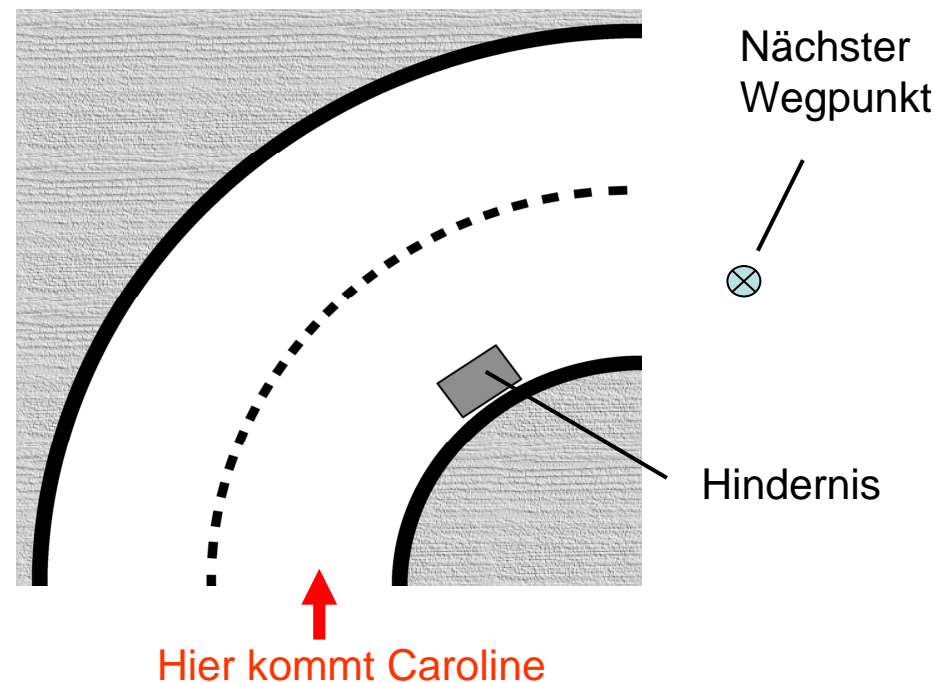
➡ Unfall mit Haus oder parkendem Auto

In der Mitte der Eigenen Spur bleiben

➡ Unfall mit statischem Hindernis

Gleicher Abstand zu allen Hindernissen

➡ Fahren auf dem Mittelstreifen

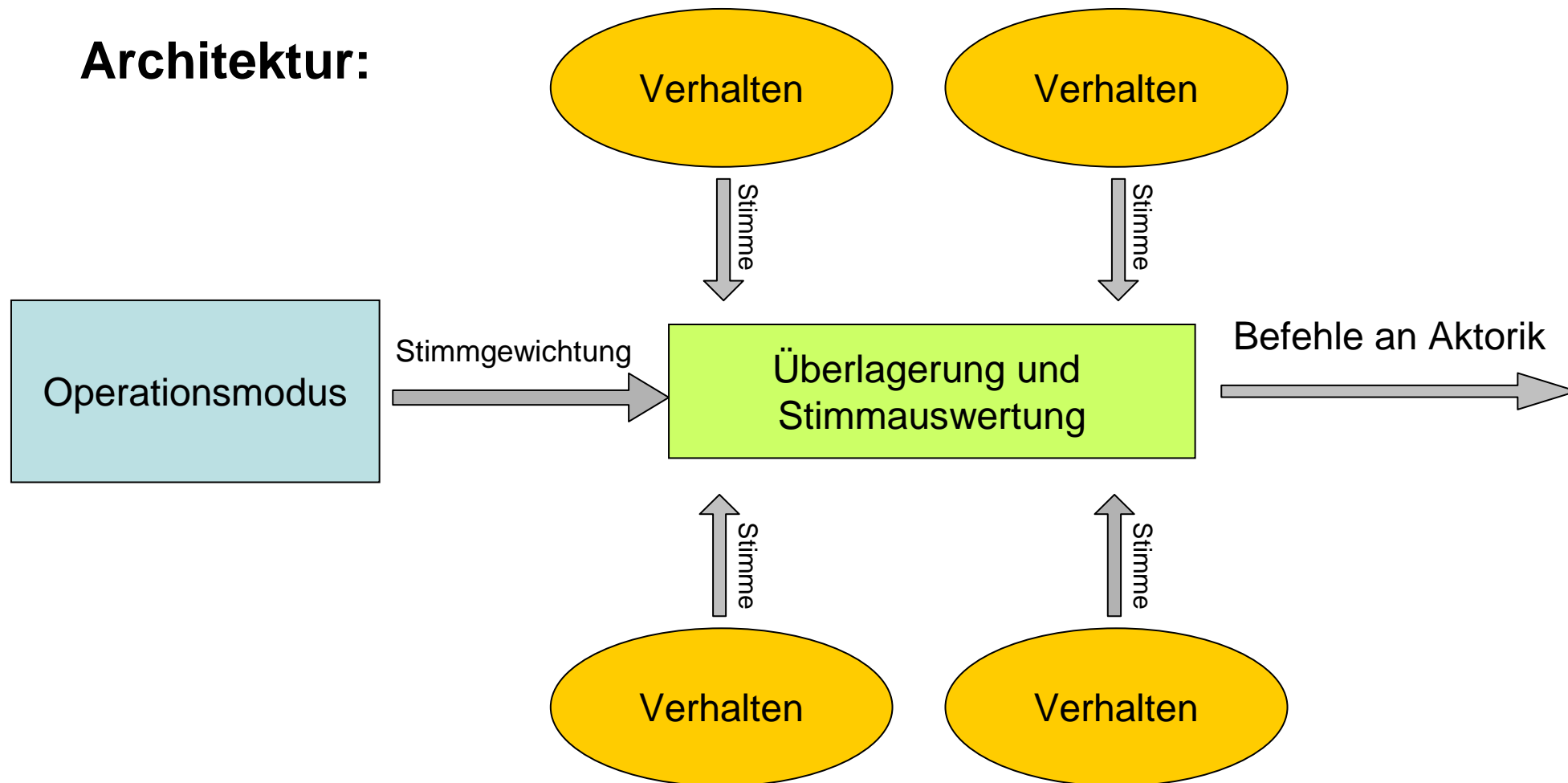


Nur das Beachten der Vorgaben aller Verhalten führt zum gewünschten Ziel.



Strategische Planung

Architektur:



Strategische Planung

Verhaltensansatz

Vorteile des Verhaltensansatzes

- Sehr gut inkrementell erweiterbar
- Problemstellung wird in Teilprobleme zerlegt
- Verschiedene Granularitäten von Verhalten möglich

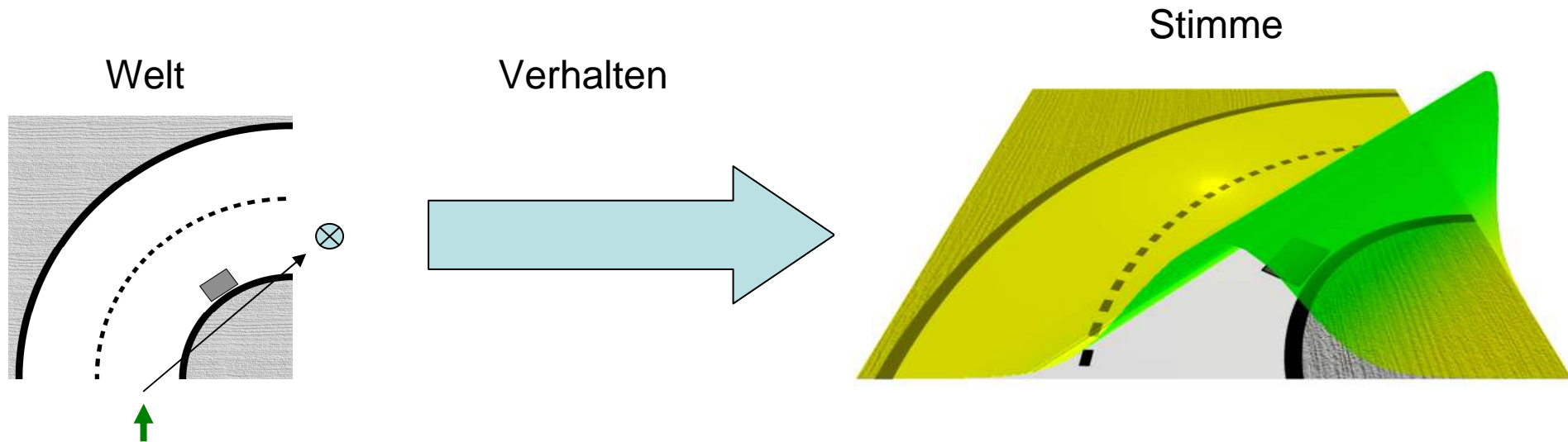
Problem: Wie arbeiten die Verhalten zusammen?

- Jedes Verhalten bewertet die möglichen Werte der Stellgröße (z.B. Lenkradstellung)
- Die Stimmen werden gewichtet und überlagert
- Gewichtung anhand von Operationsmodi wie etwa „Straßenverkehr“, „offene Zone“ oder „Stoppschild, Schlange stehen“



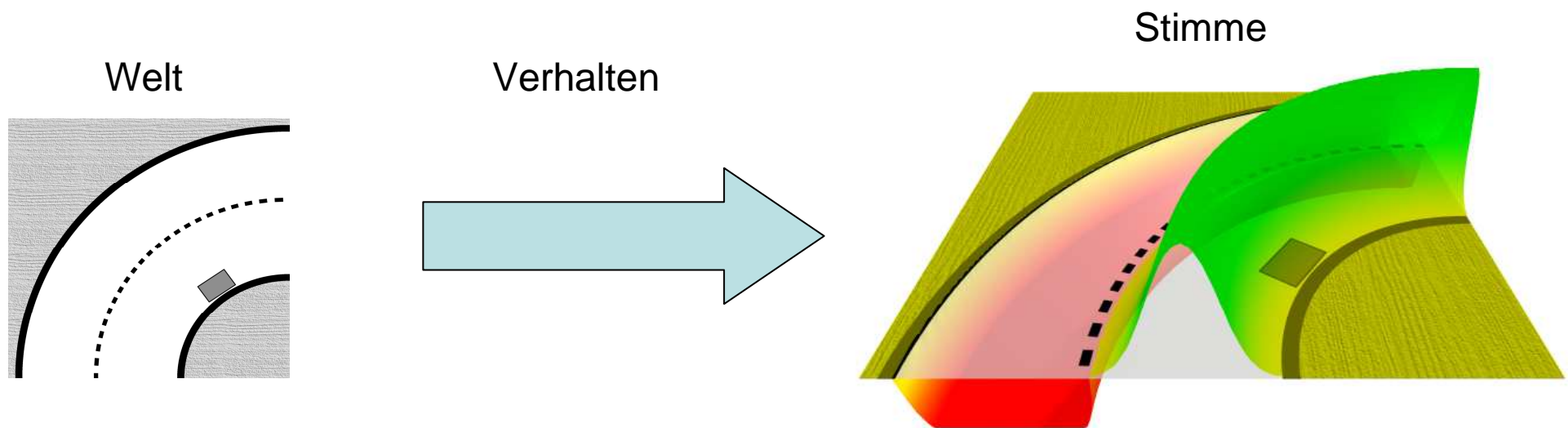
Strategische Planung

1. Den nächsten Wegpunkt anfahren



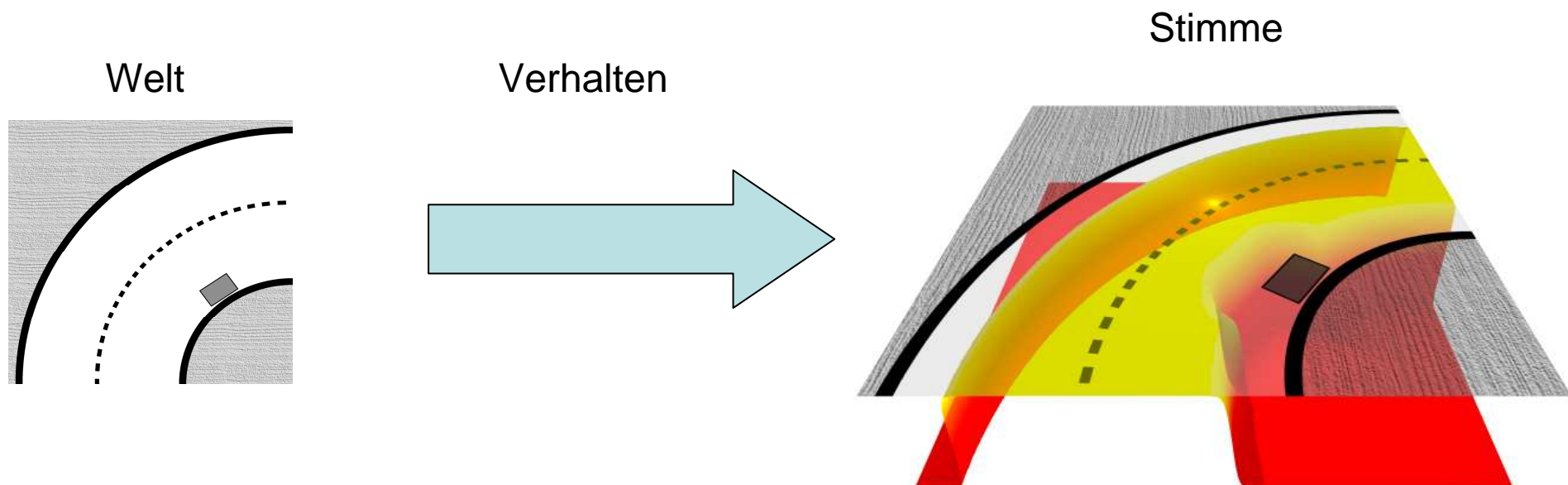
Strategische Planung

2. Der Fahrspur Folgen



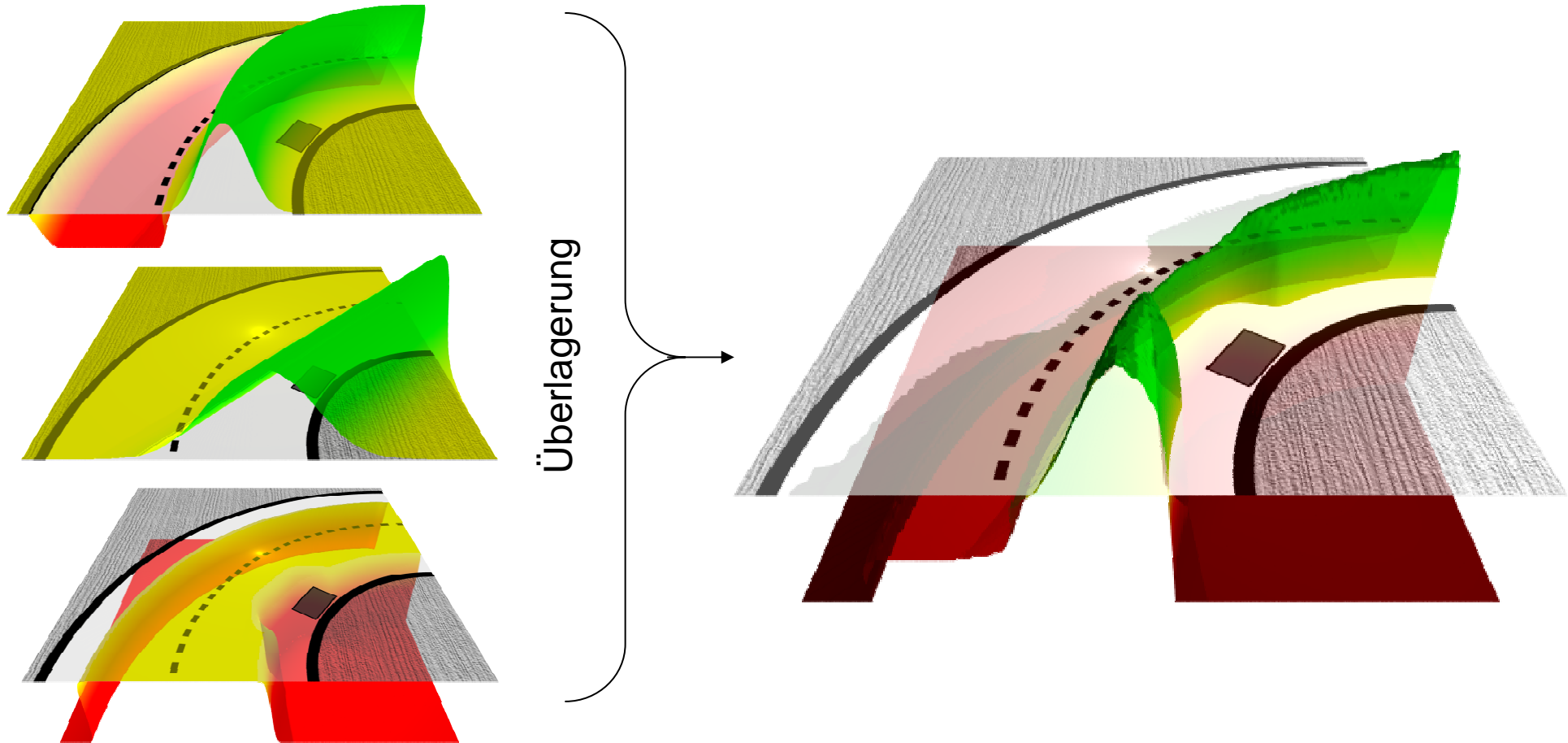
Strategische Planung

3. Hindernisse meiden



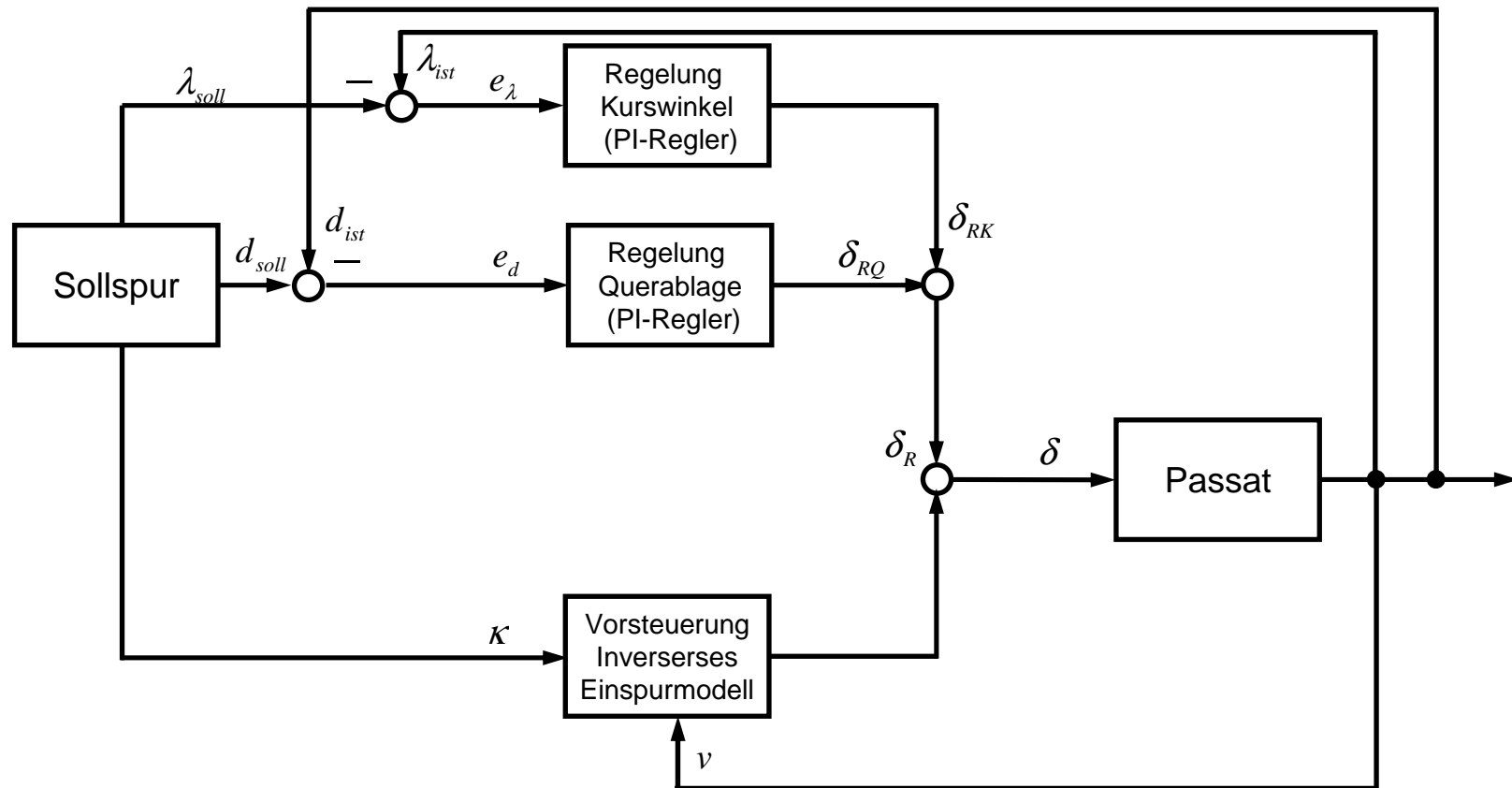
Strategische Planung

Ergebnis der Abstimmung



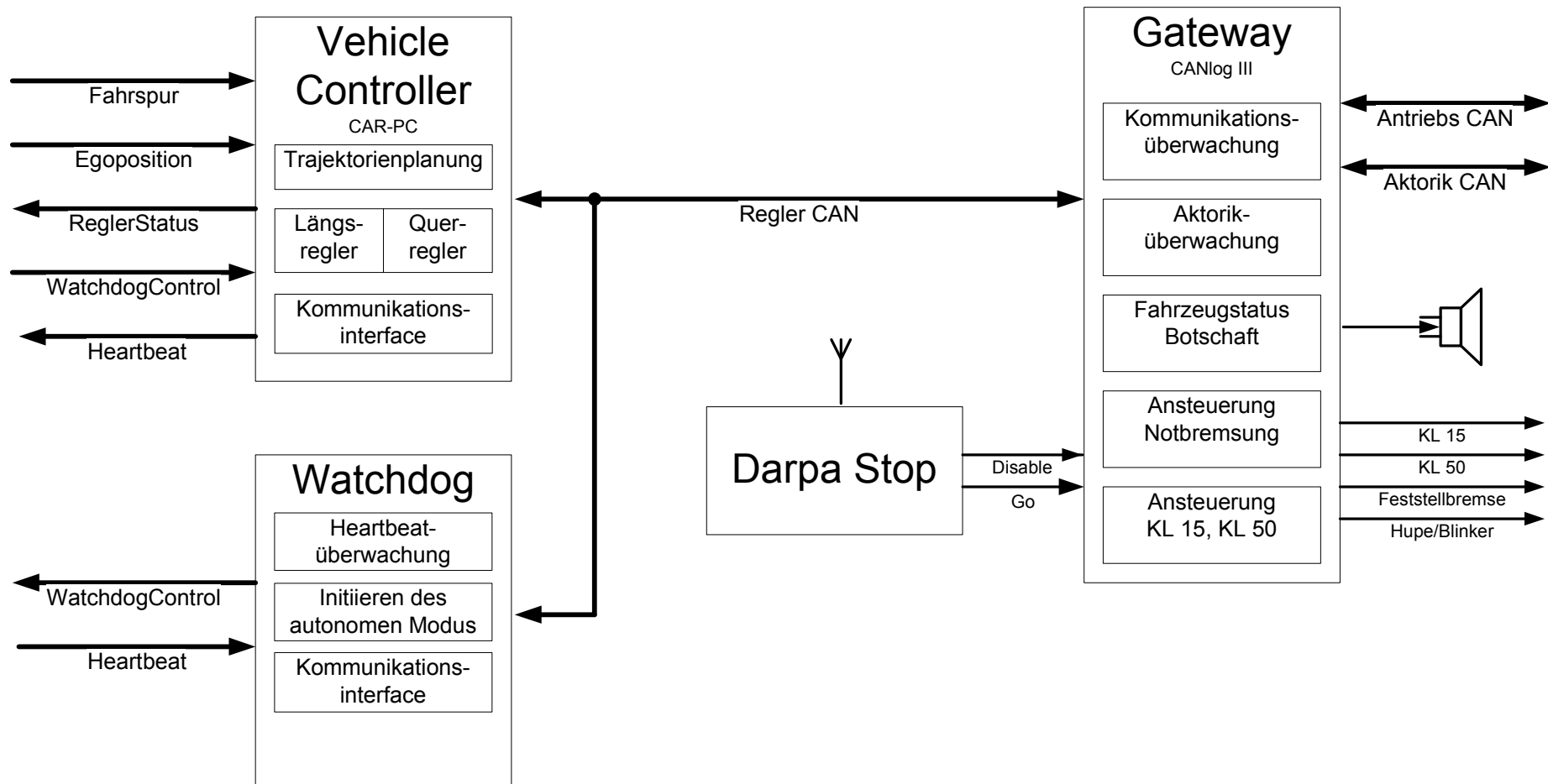
Regelung mit Vorsteuerung

Getrennte Regelung von Kurswinkel und Querablage



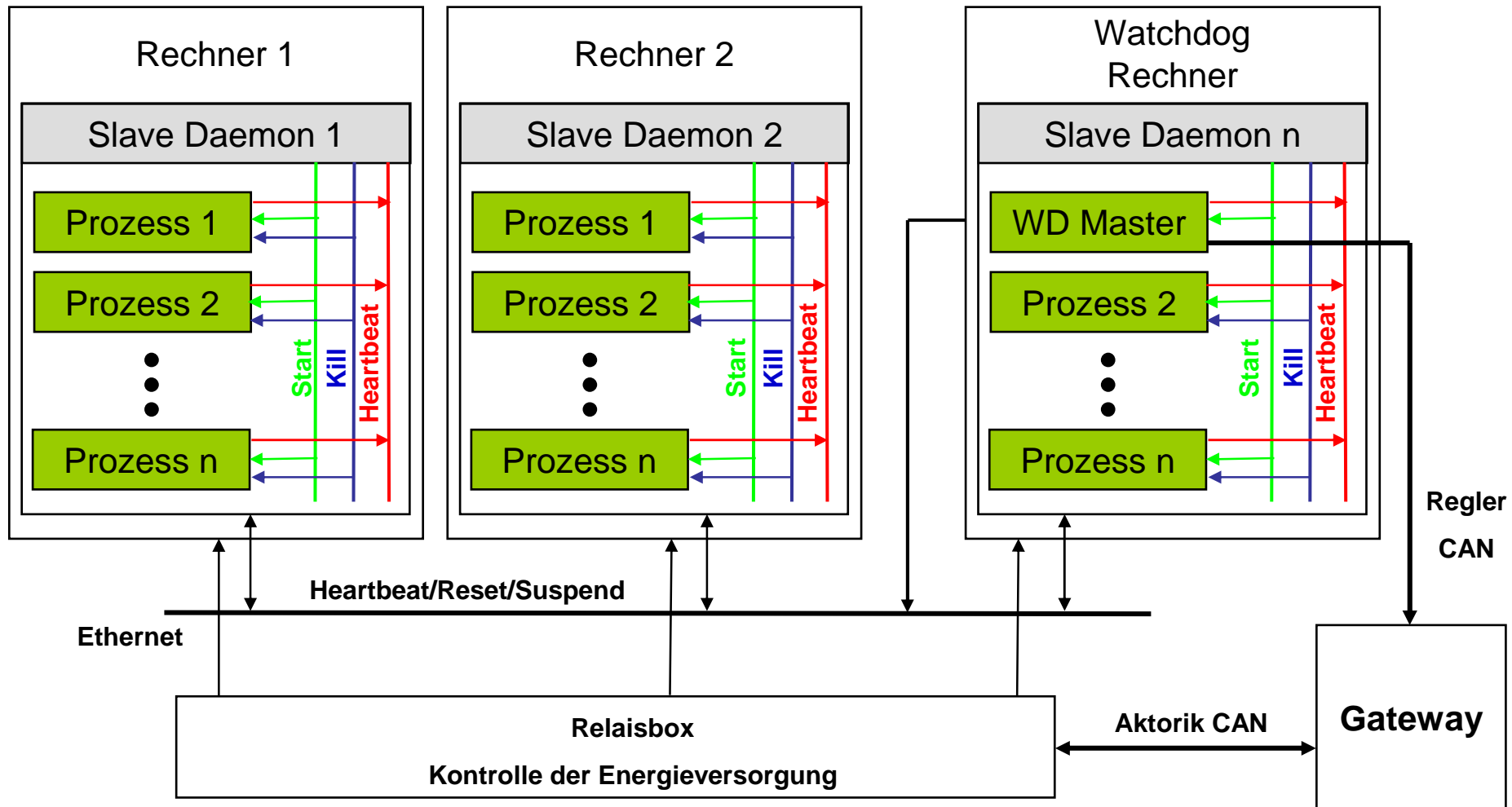
Sicherheitskonzept

Einbindung des Watchdog



Sicherheitskonzept

Heartbeatüberwachung



Rechnerarchitektur

Anforderungen

- Hardware
 - Robust gegen Hitze, Vibrationen, EMI
 - Hohe Energieeffizienz
 - Kompakte Bauformen
- Dediziertes Echtzeitnetzwerk
 - Garantierbare Untergrenze Bandbreite
 - Garantierbare Obergrenze Antwortzeit
 - Deterministisches Verhalten
- Betriebssystem
 - Verteiltes Echtzeitsystem mit gemeinsamer Uhr (Synchronität)
 - Treiberunterstützung für CAN, Kameras, GPU, IMU, ...
 - Breite Softwarebasis, Nutzung vorhandener Open Source Komponenten (z.B. NTPD, GPST, Postgres, ...)



Rechnerarchitektur

Rechner im Fahrzeug

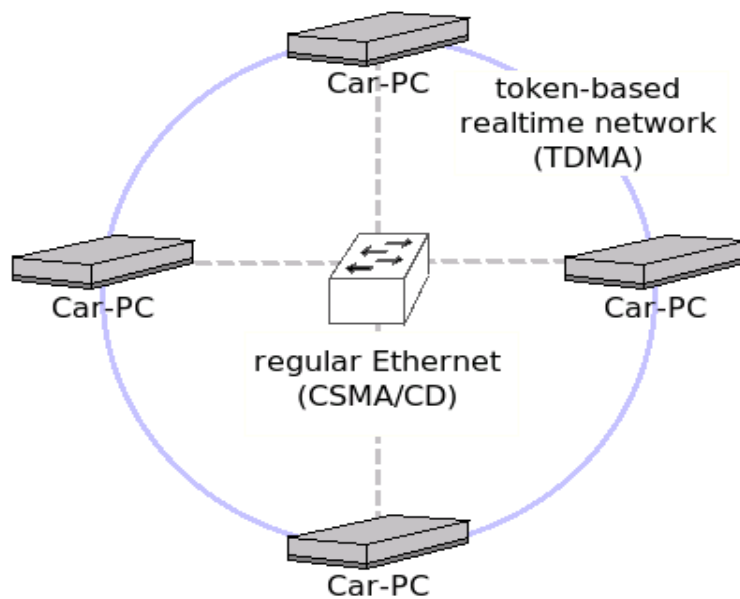
- 5x Pentium M 2,1 GHz (Bressner BT-6910)
 - Data Acquisition (2x)
 - Digitale Karte
 - Strategische Planung
 - Taktische Planung, Regler und Watchdog
- 3x Core 2 Duo 2,33 GHz (Netco OfficeCarPC)
 - Fahrspurerkennung und Bildverarbeitung
 - Kameras
 - Hochleistungs-GPU



Rechnerarchitektur

Netzwerkarchitektur

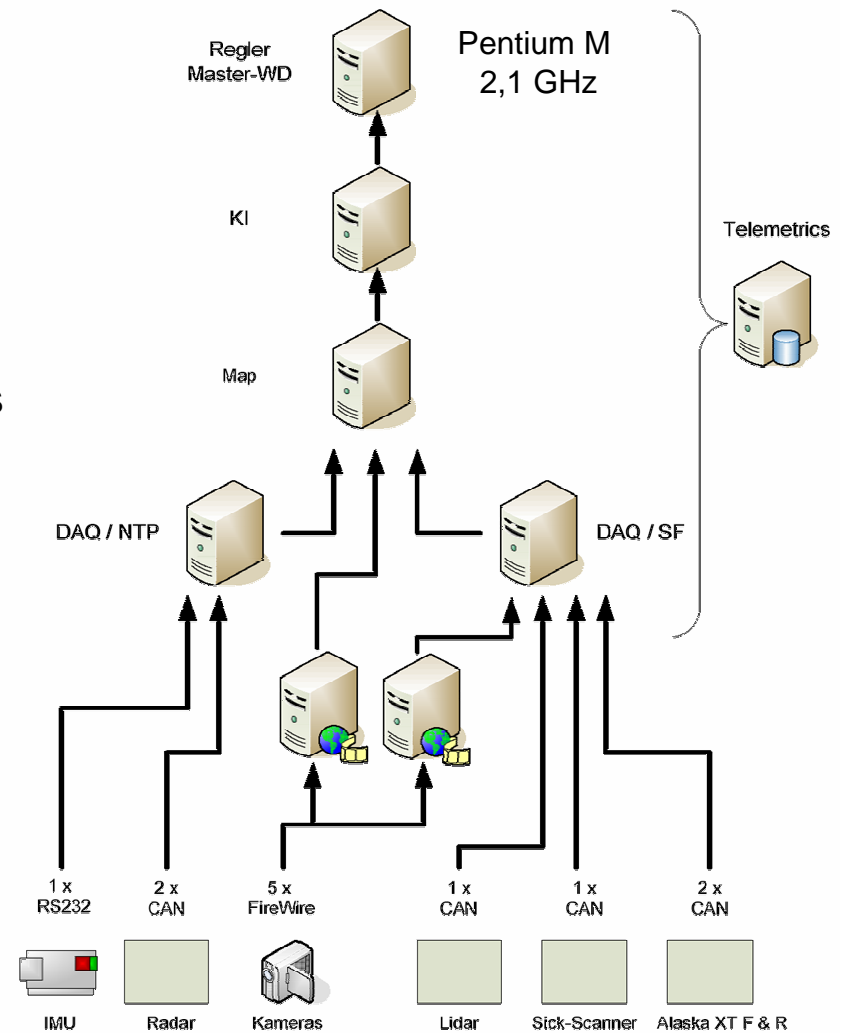
- Fast Ethernet für “normale” Kommunikation
- Dediziertes Echtzeitnetzwerk für Realtime-Messaging



tatsächliches Mapping



in
Caroline



CarOLO-Projekt
DARPA Urban Challenge 2007
TU Braunschweig



Prozess zur Sicherstellung geforderter Fahrfunktionen

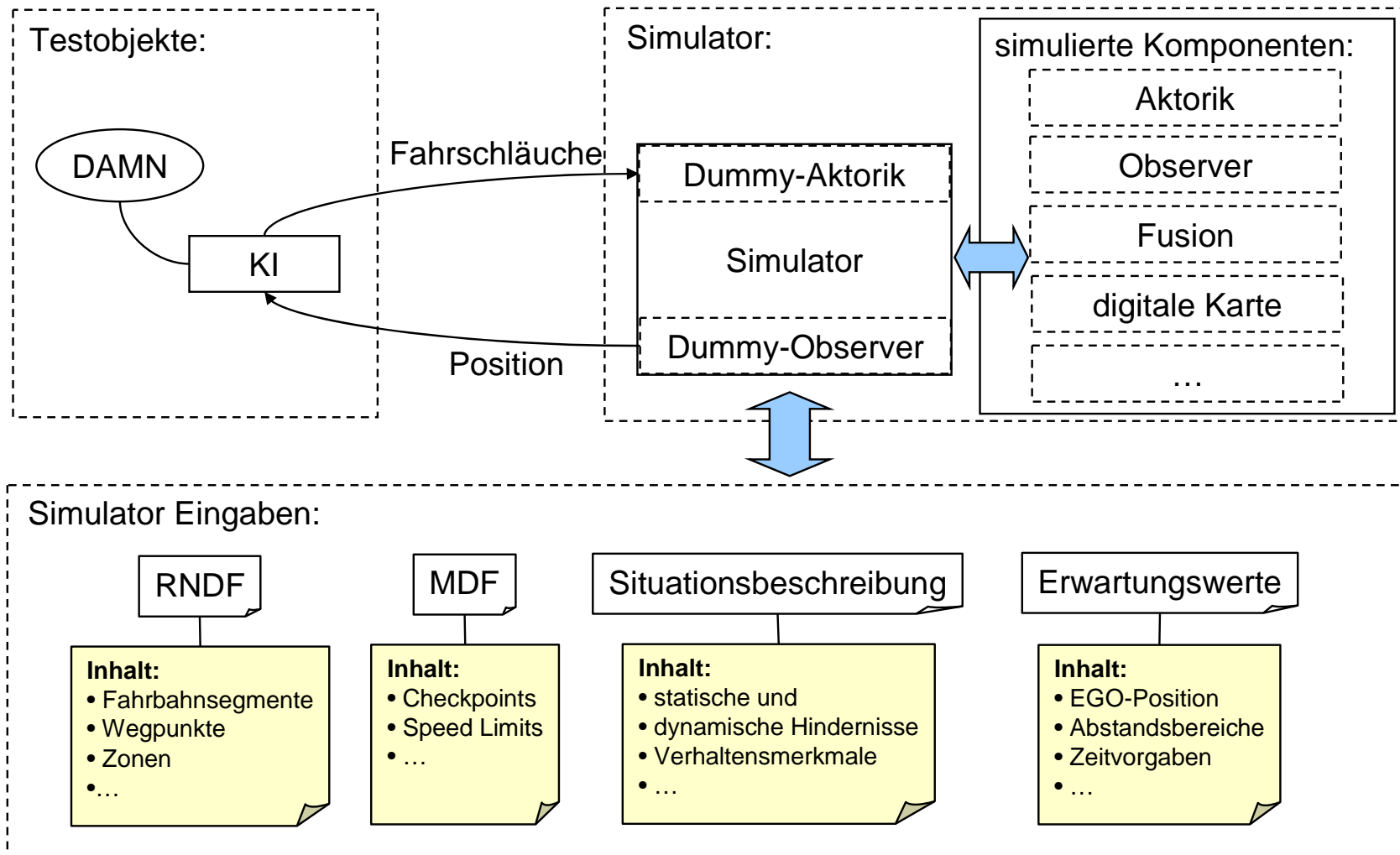
Testen und Simulation

- **virtuell**
 - automatisierte Testfälle durch den Einsatz eines Unit-Test-Frameworks
 - Aufbau einer Testinfrastruktur
 - benötigte Komponenten werden über Test-Fixtures instantiiert und miteinander verbunden
 - Testszenarien nach dem Blackbox-Verfahren
 - Vordefinierte Eingabewerte (RNDF, MDF, ...)
 - Vergleich von IST- und SOLL-Verhalten
 - Aufbau eines Testfallkatalogs
 - Simulation von Standardszenarien
 - Ziel: möglichst viele Standardsituationen durch Testfälle abdecken
 - meilensteinorientierte Gewichtung nach Schwierigkeitsgrad
- **real**
 - Definition und Aufbau von Testszenarien



virtuelle Überprüfung durch Akzeptanztests

Testen und Simulation



reale Überprüfung durch Testszenarien

Testen und Simulation

C.	Advanced Navigation	10
C.1.	Basic Traffic	10
C.2.	Obstacle field	10
C.3.	Parking lot	10
C.4.	Dynamic re-planning	12
C.5.	Road following	
C.6.	GPS outage	Parking lot

D. Advanced Traffic..... Vehicle exhibits correct parking lot behavior and demonstrates correct spot.

Extraktion von
Detailanforderungen
aus den DARPA-
Vorgaben

#	Short Description	Full Description
C.3.1	obstacle field	Das Fahrzeug erfüllt die Anforder
C.3.2	Erkennung eines Parkplatzes	Dies Software erkennt die im RNC
C.3.3	K-Rails	die Parkplätze können durch http://en.wikipedia.org/w
C.3.4	vorwärts einparken	Das Fahrzeug fährt vorwärt
C.3.5	Waypoints	Das Fahrzeug muss beide P
C.3.6	rückwärts ausparken	Das Fahrzeug verlässt die P A.7



Erarbeitung von Testszenarien



CarOLO-Projekt
DARPA Urban Challenge 2007
TU Braunschweig



Software Engineering Process

Umfeldanalyse des heterogenen Projektumfelds

– Projektmitglieder

- mehrere Institute mit Professoren und Wissenschaftlichen Mitarbeitern
- verschiedene Studiengänge
- unterschiedliches Vorwissen
- ungleiche Studiendauer

– Organisation

- Abstimmung der Institute
- Synchronisierung mit verschiedenen, institutsbezogenen Projekten
- ggf. konkurrierende Studienziele

– Hardware

- PC

– Software

- **Programmiersprachen** C, C++, Matlab/Simulink
- **Betriebssysteme** Microsoft Windows, Linux und Steuergeräte-OS
- **Entwicklungsumgebungen** Eclipse



Software Engineering Process

Anforderungen

- Ziel: **DIE SOFTWARE IST IMMER LAUFFÄHIG!**
 - agiler Software Engineering Prozess
 - mit heterogenem Umfeld
 - aggregierte Berichterstellung und zentrale Bereitstellung
 - zeitnahe Bewertung der Software Qualität
 - Codierungsrichtlinien
 - Umsetzung des „test-first“-Vorgehens
 - automatisierte und unbeaufsichtigte Berichterstellung
 - Anbindung an Versionierung- und trac-System
 - sofortiges Entwickler-Feedback



Software Engineering Process

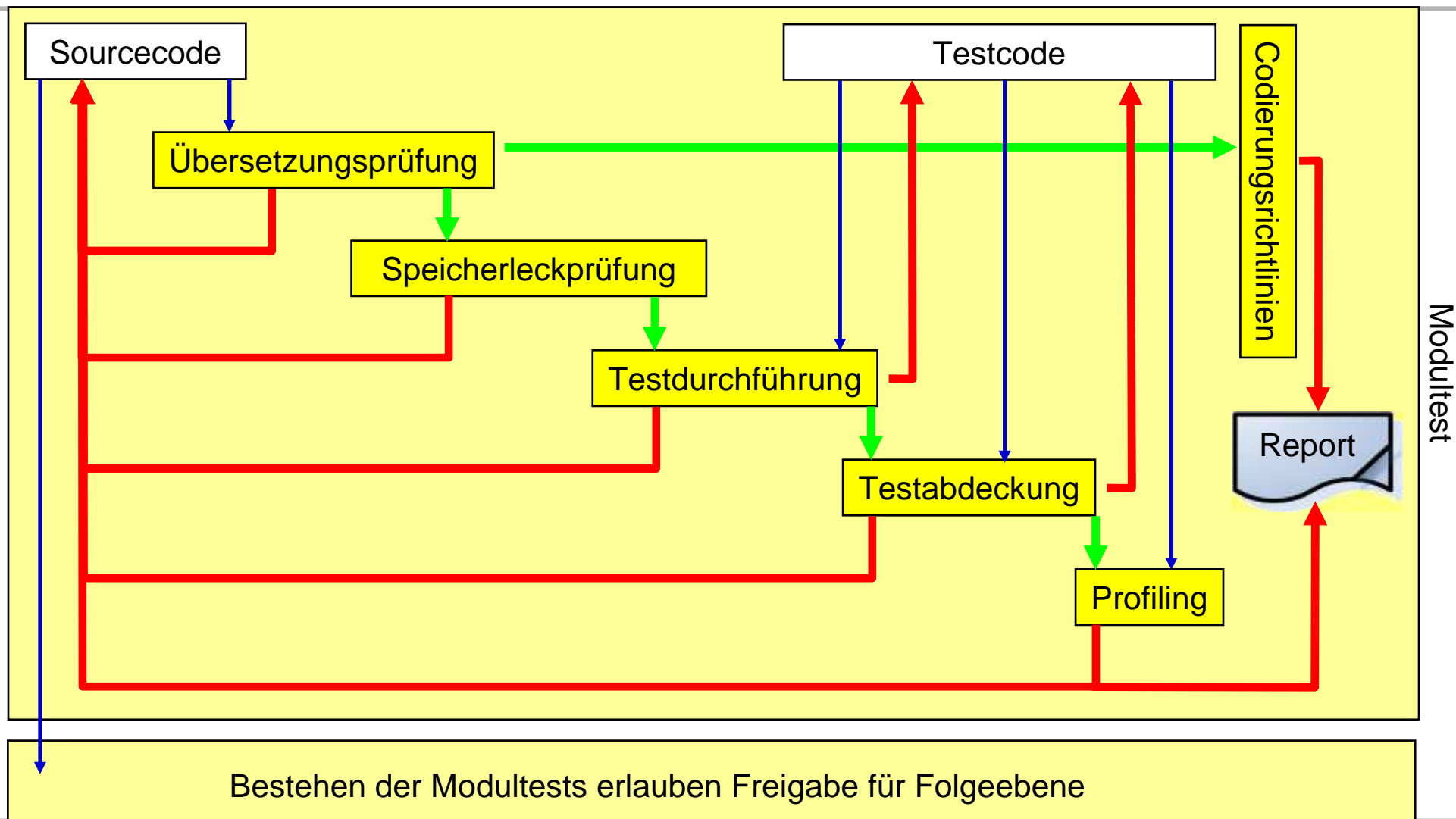
Moderner Entwicklungsprozess

- **Verzicht** auf Varianten des Quellcodes!
 - konsistenter und durchgängiger Entwicklungsprozess
 - Paradigma: **Ständig getesteter Quellcode zur fortwährenden Sicherstellung** von
 - Übersetzbarkeit
 - Einhaltung von Schnittstellen
 - permanente Integration verteilter Programmmodule
 - Ausführbarkeit der übersetzten Programme
 - **Ziel:**
 - Vertrauen in eigenen und fremden Quellcode
 - Erleichterung und Unterstützung von Refactorings
 - zeitnahe Bewertung der Software-Qualität
 - Motivation der Entwickler



Software Engineering Process

Schematische Prozessdarstellung



Application Programming Interface

Überblick SW-Komponenten

- API stellt generische Methoden zur Verfügung und abstrahiert die reale Hardware und Systemprozesse
- Namensraum für die verschiedenen Module
 - **actorics** – kapselt alle Module zur direkten Kontrolle des Fahrzeugs
 - **ai** – kapselt alle Planungsalgorithmen
 - **canobjects** – zuständig für alle via CAN gesendeten Daten
 - **cgobjectdetection** – stellt Algorithmen für kameraunterstützte Objekterkennung zur Verfügung
 - **data** – beinhaltet alle Datenklassen für die Datenübertragung zwischen Modulen (Teil der API)
 - **digitalmap** – verwaltet und erweitert ggf. die digitale Karte
 - **fusion** – Sensordatenfusion
 - **lanedetection** – kapselt die kameragestützte Spurerkennung
 -

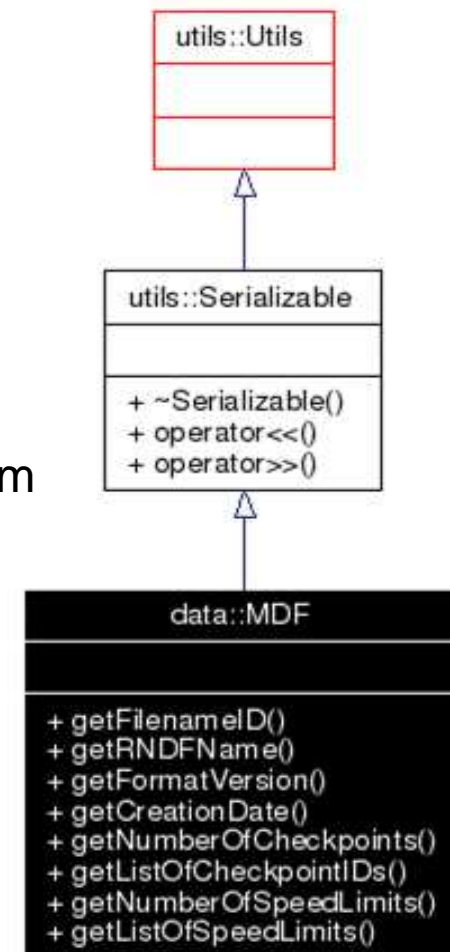
actorics
ai
boost::spirit
canobjects
cgobjectdetection
controlelements
data
digitalmap
exceptions
fusion
lanedetection
observer
recorder
sensors
simulator
std
ui
utils
watchdog



Application Programming Interface

Beispiel: Missions Daten Datei (MDFILE)

- enthält alle Datenklassen für zwischen Modulen ausgetauschten Objekten
- bietet generische Eigenschaften für alle Objekte
 - Serialization
 - Deserialization
 - Konvertierung und Darstellung des Inhalts in lesbarer Form
- Beispiel: MDF
 - repräsentiert richtig gepackte Datei mit Missionsdaten
 - bietet ein Interface zum Einlesen der Daten
 - aus Klasse MDFFILE gebildet



Caroline – ein autonom fahrendes Fahrzeug im Stadtverkehr

Zusammenfassung und bisherige Erfahrungen

- robuste Rechnerhardware erforderlich mit hohen Anforderungen an Stabilität und Verfügbarkeit des Betriebssystems
- einheitliche Schnittstellen und eine saubere Architektur für alle Module erlauben einfache Wartbarkeit und Austauschbarkeit
- Projektarbeit mit großem interdisziplinärem Team verbunden mit einem engem zeitlichem Rahmen in einem universitären Umfeld stellt hohe Anforderungen an Teammitglieder, -prozesse und Werkzeuge

